

Neural Networks in Control

neurocomputers are here

By Owen Bishop

Most of the actions of most animals are controlled by a network of nerve cells, or *neurons*. In this article Owen Bishop looks at the possibilities of implementing the neurocomputer.

Neurons are usually organised into well-defined structures, such as the brain, the spinal cord and the peripheral nerves. Each neuron is a centre of information transfer (Note: transfer, not storage). It can receive information as electrical signals from a number of other neurons in the network, process this information, and pass it on to other neurons. In the human brain, for instance, a neuron connects to as many as 10,000 other neurons. The extent to which information is transferred from neuron to neuron depends on the number of

physical connections existing between the neurons and the intensity (or strength) of each connection. The number of possible connections is far greater than the number of neurons present. We estimate that a human brain may contain as many as 10^{16} connections, an enormous resource for memory and intelligent action.

Learning

A young animal begins life with a given set of network connections

built in. It soon adds to these connections by experience as it learns useful responses to a range of outside stimuli. Learning involves not only making totally new connections, but also by strengthening some existing connections and weakening others. Learning is thus an essential feature of a neural network. The animal learns how to act in a given set of circumstances. It also learns patterns, as when we learn to recognise friend or foe, or to identify the smell of a well-barbecued steak. Parts of the network are sites of higher thought processes.

Given that the neural network of an animal is capable of complex recognition and control actions, researchers have tried to build the artificial equivalent, an **artificial neural network** or **ANN**. An ANN consists of neuron units mimicking the action of animal neurons. They have input, processing and output stages. The earliest electronic ANNs were based on valve circuitry, but little progress was made until the advent of VLSI. Now, with the enormous computing power of even the smallest of bench-top computers, we model ANNs mathematically. This can be done in a system based on conventional microprocessors, as illustrated by the software described later. There are also custom-designed processors used to build true **neurocomputers**. These are designed to excel in the particular kinds of processing that are needed to model ANNs, which is done most efficiently by using parallel processing.

It is important to realise that, although ANNs are built from neurons (electronic or

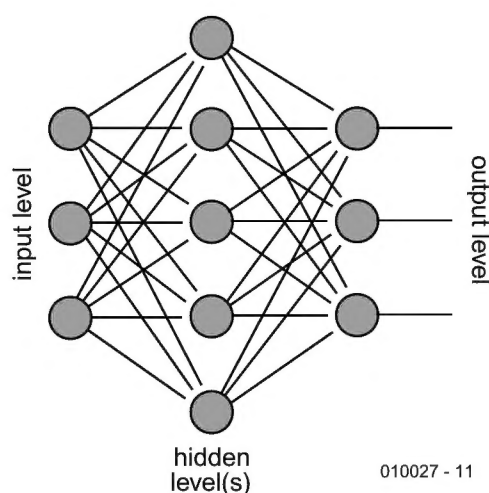


Figure 1. An artificial neural net consists of at least three levels of connected neurons.

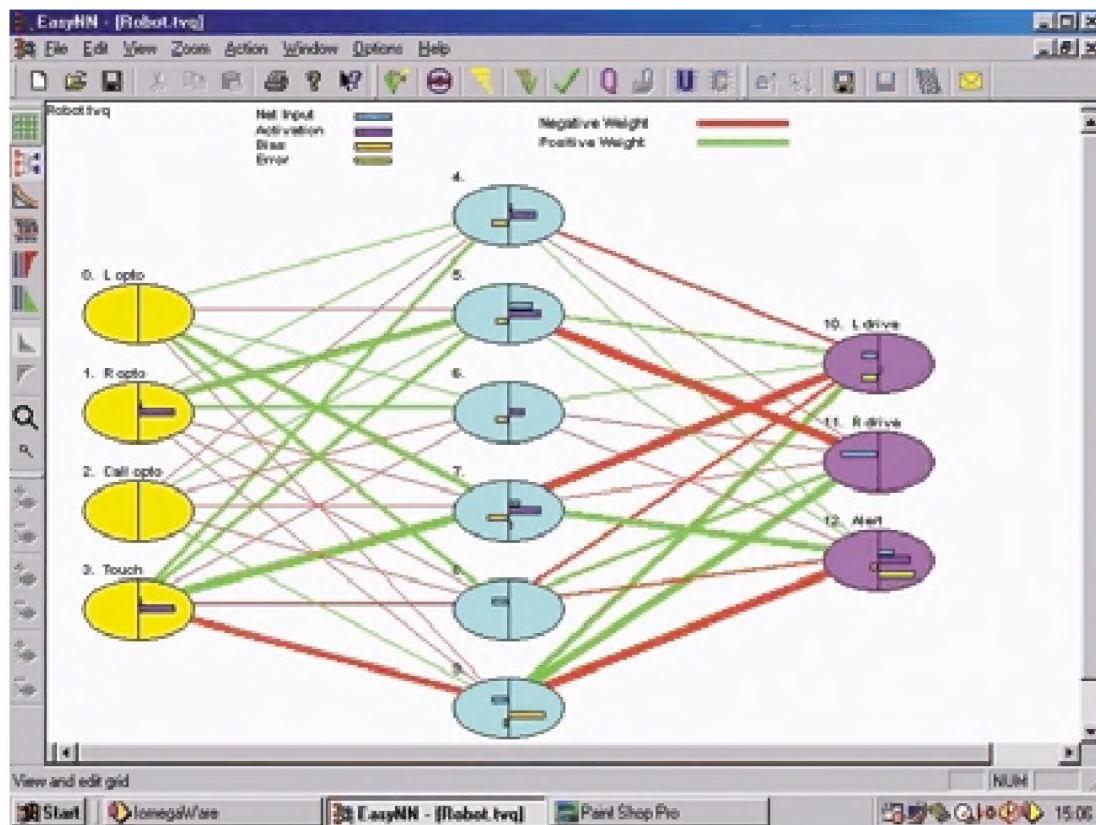


Figure 2. This ANN models the control system of a platform robot.

mathematical), the action of which superficially resembles that of animal neurons, the analogy goes no further. We know a lot about the structure and activities of animal neurons but very little about how they interact. A neurocomputer operates in a way that is probably very different from the operation of a brain. Nevertheless, it can perform similar functions.

Architecture

The structure of a typical ANN (electronic or mathematical) is illustrated in **Figure 1**. It consists of at least three layers (or levels) of neurons, and there may be any number of neurons in each level. The input level consists of neurons that receive input from sensors. In a control system, these might include thermal, optical, proximity or pressure sensors. In a visual pattern recognition system, the inputs might be from specified areas in the field of a video camera. The **output level** consists of the neurons that drive actuators, such as motors, solenoids and lamps. On the other hand, maybe they generate bits in a digital code that is then interpreted by a computer. Between the input and output levels are one or more **hidden levels**, which are not directly connected to the outside world.

In the network of Figure 1, signals pass in

one direction from the input level, through the hidden levels, to the output level. This is known as **feedforward architecture**, and is the one most commonly used. Certain applications may have other architectures involving feedback, connections between neurons on the same level, and other configurations.

Modelling an ANN

A number of computer programs are available for modelling ANNs, including a PC program known as **Easy NN**, by Stephen Wolstenholme. To study an ANN in action, we set up a control network for a hypothetical platform robot. The robot circulates the workstations in a factory, to collect completed workpieces and to deliver spare parts. Its route is determined by a white line painted on the floor. The line is a complete loop with branches to each of the workstations. The robot straddles the line and has two optical sensors at the front, which scan the floor to the left and right of the line. When the robot is moving along the line, neither of the sensors detects the line and both

its drive motors are running. However, if it deviates from the line, or where the line bends, one sensor or the other is stimulated and the appropriate drive motor is switched off to cause the robot to return to the line. When there is a branch line, the extra width of the tracks stimulates both sensors at the same time. Then the robot may ignore the main track and branch off to the left toward the workstation, but it only does this if its 'call' optical sensor is receiving a beam of light directed at the robot from the workstation.

The robot also has a touch sensor at the front, to detect a barrier when it reaches the workstation. Then it sounds an alert and stops while items are unloaded or loaded, until the barrier is moved aside. It then continues along the branch track until it rejoins the main track. The touch sensor also detects obstacles it may encounter. Then it stops and sounds the alert until the obstacle is removed.

This is an oversimplified control system that would probably not work well in practice, but it serves to illustrate the nature of an ANN.

Another point is that this network has only binary inputs and outputs. However, mathematical models of ANNs are perfectly capable of working with analogue values.

Training

The first stage in setting up the network using **Easy NN** is to specify that there are to be four input neurons, three output neurons, and a single level of hidden neurons. The software was left to decide how many hidden neurons there should be. Next, the network was taught how to behave by keying in a table, showing a selection of possible input values and what output values are expected in each case. Since this is a binary example, we keyed in all sixteen possible combinations of the four inputs at levels 0 (false) and 1 (true). We also keyed in the expected output responses for each combination of inputs. We simply decided what the robot is to do and keyed in '0's and '1's accordingly. This is known as training by **supervision** and is the equivalent of programming an expert system. Because the system is working on binary values, the table is the equivalent of a truth table.

When the computer is told to generate the network, it first decides how many neurons are needed in the hidden layer. Then it connects all the neurons (as in Figure 1) and gives each connection random but relatively low weighting. It then puts in the combinations of inputs, computes the corresponding outputs and compares them with the expected outputs. This process is repeated and, because each neuron has a memory associated with it, the weightings can be adjusted to produce an output that more closely matches the expected output. The processing continues until stable values are obtained. The network is then displayed as a diagram (**Figure 2**). From this, we can see that the computer has elected to have six hidden neurons. The inter-neuron connections are shown as lines of varying width, depending on the weightings. Some of these are displayed in green to indicate positive (excitatory) action. The others are in red to indicate negative (inhibitory) action. The signals arriving at each

neuron are summed by the neuron in proportion to the weightings. In most systems a number of weak signals arriving at a neuron are ignored. Only when the total of signals exceed a given threshold, does the neuron generate an output signal.

Figure 2 represents one particular combination of inputs and the corresponding outputs. Parameters at each neuron are shown by bar charts of which the second bar down represents the activation state. To investigate the behaviour of the network (and the robot) we call up a 'Query' panel and key in the level of each sensor signals (0 or 1). The resulting output is automatically calculated and a version of Fig. 2 with the new inputs and outputs is displayed. In Fig. 2, for example, the activation states have been set so that the right opto-sensor and the touch sensor are both '1'. The left opto-sensor and call sensors are both '0'. The corresponding output is '0' for both drives and '1' for the alert. The robot has met an obstacle on its path, has stopped and is sounding the alert. If we alter the inputs to make all opto-sensors '1' and the touch sensor '0', the output instantly changes to left drive '1' (ON), right drive '0' (OFF) and alert '0' (OFF). The robot is turning right at a junction. Trying various combinations of inputs always gives the result expected according to the training that was given.

This simple controller could have been built from three or four hardware logic gates, or programmed as a few lines of logic commands in a BASIC program. On the other hand, we could have programmed the 'truth table' as a lookup table and searched it to find the correct outputs. Instead, we made use of algorithms that describe the action of a neuron and what happens when several such neurons are connected in a network. The only causal connection between input and output is through the ANN. Incidentally, the network does only what we have trained it to do. If we train it to respond differently, it will learn the new pattern of behaviour.

Analogue input

One of the examples available at the **Easy NN** site shows how well an ANN can cope with analogue

input. The network is trained to identify a specimen of one of three species of *Iris*, given only the length and width of the petals and sepals. The network has four input neurons, one for each of the four input quantities. It has three output neurons, one for each of the species, *I.setosa*, *I.versicolor* and *I.virginica*. The computer generates seven neurons in the hidden layer.

With analogue input, the network can not be trained by tabulating all possible combinations of inputs. Instead, it is given a sample of analogue data to work on. The four measurements taken from each of a sample of flowers are keyed in to the data table. 'True' is entered in one of the three output columns to indicate to which species each set of data refers. 'False' is entered in the other two columns. The table holds data from 50 flowers of each species, which is considered sufficient for training.

Although the correspondence between dimensions and species is by no means easy to distinguish by inspection, the network quickly learns to identify the species of other specimens of *Iris* of these three species from these four basic measurements.

A similar but much larger network is able to distinguish between sixteen species of cichlid fish, using 10 input neurons receiving analogue data on body features.

Applications

Given a certain pattern of input, an ANN learns to produce an appropriate pattern of output. This makes ANNs. We have shown examples in the fields of control and taxonomy. ANNs are often used for scanning masses of data looking for patterns. It is reported that data on credit card fraud revealed that fraud is especially common among young women buying shoes. Visual recognition is important in security systems and robotics. ANNs are finding many applications in these fields. The flexibility of ANNs and their ability to respond to changes in input make them useful in tuning the parameters of fuzzy logic systems. In the future, we may find that many devices in the home are controlled by a combination of fuzzy logic and an artificial neural network.

Software

At the time of writing, full details of **Easy NN** are published at

<http://www.tropheus.demon.co.uk/>.

(010027-1)